UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/731,632	12/09/2003	Ashutosh K. Jha	NVDA P001157	4721

26291 7590 12/24/2008

PATTERSON & SHERIDAN L.L.P. NJ Office
595 SHREWSBURY AVE, STE 100
FIRST FLOOR
SHREWSBURY, NJ 07702

EXAMINER	
ROSE, KERRI M	

ART UNIT	PAPER NUMBER
2416	

MAIL DATE	DELIVERY MODE
12/24/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/731,632	JHA ET AL.	
	Examiner	Art Unit	
	KERRI M. ROSE	2416	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 December 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,8-12,15-17,19,20 and 26-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5,8-12,15-17,19,20 and 26-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Response to Arguments

1. Please note AU 2616 is now AU 2416.

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 10/01/2008 has been entered.

3. Applicant's arguments, see page 7, filed 12/2/08, with respect to the rejection(s) of claim(s) 1-5, 8-12, 15-17, and 19-21 under 103 have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of new reference Allison et al. (US 6,393,457).

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-5, 8-12, 15-17, and 19-21 are rejected under 35 U.S.C. 103(a) as being obvious over Hayes (US 2003/0158906) in view of Connery et al. (US 6,246,683) further in view of Allison et al. (US 6,393,457).

In regards to claim 1, Hayes discloses a method of processing frames for a TCP connection comprising: processing a first portion of the frames using an offload unit to produce first processed frame data (figs. 7 and 12 disclose an offload processor. Paragraph 38 discloses offloading a portion of the frames for processing); processing a second portion of the frames using the offload unit to produce second processed frame data (figs. 7 and 12 disclose an offload processor. Paragraph 38 discloses offloading a portion of the frames for processing); and processing the second processed frame data using the TCP stack executed on a CPU to produce third processed frame data (Paragraph 39 discloses sending some frames back to the CPU for further processing using the TCP stack if the offload processor cannot complete the processing.)

Hayes is silent to uploading the first processed frame data to a user buffer in a first portion of memory that is allocated to an application program; and uploading the second processed frame data to a legacy buffer in a second portion of the memory that is allocated to a software driver configured to communicate between the offload unit and a TCP stack. Additionally, Hayes is silent to receiving a user buffer descriptor specifying a location of a plurality of user buffers in a first portion of memory that is allocated to an application program, wherein each user buffer is configured to store processed frame data for the delegated TCP connection; and indicating in a filed of the user buffer descriptor how many of the user buffers store the first processed frame data for the delegated TCP connection.

Connery discloses uploading the first processed frame data to a user buffer (fig. 4 element 111 discloses a buffer) in a first portion of memory that is allocated to an application program (fig. 4 indicates the buffer, 111, is located in a higher layer managed memory. Col. 6 lines 41-44 disclose it is an application managed memory.) and uploading the second processed

frame data to a legacy buffer (fig. 4.110 discloses a buffer) in a second portion of memory that is allocated to a software driver (fig. 4 indicates the buffer, 110, is located in a driver managed memory) configured to communicate between the offload unit and a TCP stack (fig. 3 discloses the driver layer, 54, communicates between the offload unit, 56, and a TCP stack, 52.). Connery discloses passing identification of a target buffer as a buffer descriptor including an address and length in col. 5 lines 1-6.

It would have been obvious to one of ordinary skill in the art at the time of the invention to upload the first and second portion to buffers in application or driver memory respectively, as taught by Connery, in the offloading method taught by Hayes because doing so improves the performance and scalability of a network, as taught by Connery in col. 3 lines 4-10.

Allison discloses a buffer count field in a user buffer descriptor in fig. 3 and col. 5 lines 51-60. The field indicates the number of buffers which contain data.

It would have been obvious to one of ordinary skill in the art at the time of the invention to include a buffer descriptor with count field, as taught by Allison, in the offload method taught by Hayes because doing so helps provide architecture capable of speeds from 10-1000 Mbps, as taught by Allison in col. 2 lines 15-17.

In regards to claim 2, Hayes and Connery disclose the method of claim 1, further comprising determining whether a special case exists during the processing of each of the frames (Hayes p. 38 discloses determining if an "exceptional condition", in other words a special case, exists during processing.)

In regards to claim 3, Hayes and Connery disclose the method of claim 1, wherein at least a portion of the first processed frame data is payload data (Connery fig. 4.102 discloses the first portion consists of payload data.)

In regards to claim 4, Hayes and Connery disclose the method of claim 1, wherein at least a portion of the second processed frame data is partially processed frame header data (Connery fig. 4.101 discloses the second portion consists of header data.)

In regards to claim 5, Hayes and Connery disclose the method of claim 1, wherein at least a portion of the second processed frame data is payload data (Connery fig. 4.101 is termed a header fragment. However, it is disclosed the fragment is made up of Ethernet, IP, TCP, and SMB headers. As each header is added it is common to consider the remainder of the packet as payload. In other words, the Ethernet header encapsulates a payload which also happens to include IP, TCP, and SMB headers.).

In regards to claim 8, Hayes and Connery disclose the method of claim 1, further comprising finishing processing of the second processed frame data by the TCP stack executed on the CPU (Hayes p.39 discloses finishing processing by the TCP stack on the CPU if the offload processor cannot complete processing. Connery fig. 3 and col. 6 line 58 – col. 7 line 20 disclose completing processing of the second processed frame conventionally, using the TCP stack.).

In regards to claim 9, Hayes discloses a system for processing data for a TCP connection, comprising: a TCP stack (fig. 10.118 discloses a conventional TCP stack) configured to process received frames (p. 39 discloses processing certain receiving frames using the conventional TCP stack); a software driver configured to interface between the TCP stack and an offload unit (fig.

10. 116 and 160 are device drivers which interface between the TCP stack and offload unit.); and the offload unit (fig. 10.26c is the offload unit) configured to: process frames received on a delegated connection to produce payload data and partially processed frames (figs. 7 and 12 disclose an offload processor. Paragraph 38 discloses offloading a portion of the frames for processing).

Hayes is silent to a memory configured to store user buffers in a first portion that is allocated to an application program and to store legacy buffers in a second portion that is allocated to the software driver; and uploading partially processed frames to at least one of the legacy buffers; and upload the payload data to at least one of the user buffers. Additionally, Hayes is silent to receiving a user buffer descriptor specifying a location of a plurality of user buffers in a first portion of memory that is allocated to an application program, wherein each user buffer is configured to store processed frame data for the delegated TCP connection; and indicating in a filed of the user buffer descriptor how many of the user buffers store the first processed frame data for the delegated TCP connection.

Connery discloses uploading the first processed frame data to a user buffer (fig. 4 element 111 discloses a buffer) in a first portion of memory that is allocated to an application program (fig. 4 indicates the buffer, 111, is located in a higher layer managed memory. Col. 6 lines 41-44 disclose it is an application managed memory.) and uploading the second processed frame data to a legacy buffer (fig. 4.110 discloses a buffer) in a second portion of memory that is allocated to a software driver (fig. 4 indicates the buffer, 110, is located in a driver managed memory) configured to communicate between the offload unit and a TCP stack (fig. 3 discloses the driver layer, 54, communicates between the offload unit, 56, and a TCP stack, 52.). Connery

discloses passing identification of a target buffer as a buffer descriptor including an address and length in col. 5 lines 1-6.

It would have been obvious to one of ordinary skill in the art at the time of the invention to upload the first and second portion to buffers in application or driver memory respectively, as taught by Connery, in the offloading method taught by Hayes because doing so improves the performance and scalability of a network, as taught by Connery in col. 3 lines 4-10.

Allison discloses a buffer count field in a user buffer descriptor in fig. 3 and col. 5 lines 51-60. The field indicates the number of buffers which contain data.

It would have been obvious to one of ordinary skill in the art at the time of the invention to include a buffer descriptor with count field, as taught by Allison, in the offload method taught by Hayes because doing so helps provide architecture capable of speeds from 10-1000 Mbps, as taught by Allison in col. 2 lines 15-17.

In regards to claim 10, Hayes and Connery discloses the system of claim 9, wherein the offload unit is configured to process frames for which a special case does not exist (Hayes p. 39 discloses returning frames from the offload to the main processor if a special case exists. In other words the offload unit only processes frames for which a special cases does not exist.).

In regards to claim 11, Hayes and Connery disclose the system of claim 9, wherein the offload unit is configured to notify the TCP stack when a special case is determined to exist (Hayes p. 39 discloses transferring control back to the TCP stack, from the offload processor, if a special case is determined to exist. The transfer of control serves as notice to the TCP stack of the special case.).

In regards to claim 12, Hayes and Connery discloses the system of claim 9, wherein the TCP stack is configured to process frames for which a special case exists (Hayes p. 39 discloses returning frames from the offload to the TCP stack if a special case exists.).

In regards to claim 15, Hayes discloses receiving additional frames while uploading processed frames. Frames will continue to be received unless the input buffer is full. Only then will incoming packets be denied, i.e. dropped. Therefore it is inherent that the offload processor will continue to receive frames that will wait processing while it completes processing of its current frame.

In regards to claim 16, Hayes discloses a method of processing frames for delegated and nondelegated TCP connection, comprising: processing delegated TCP connections using an offload unit (figs. 7 and 12 disclose an offload processor. Paragraph 38 discloses offloading a portion of the frames from a delegated connection for processing.) for which special cases do not exist (Hayes p. 39 discloses returning frames from the offload to the main processor if a special case exists. In other words the offload unit only processes frames for which a special cases does not exist) to produce processed frame data; processing non-delegated TCP connections using a TCP stack executing on a CPU (p. 67 discloses there are some processes, such as application specific initialization which should not be processed by the offload unit but instead processed conventionally.); processing all frames for which special cases exist using the TCP stack executing on the CPU (Hayes p. 39 discloses returning frames from the offload to the TCP stack if a special case exists.).

Hayes does not disclose uploading the processed frame data to a user buffer in a first portion of memory that is allocated to an application program; and uploading frame data for the

non-delegated TCP connection to a legacy buffer in a second portion of the memory that is allocated to a software driver configured to communicate between the offload unit and the TCP stack. Additionally, Hayes is silent to receiving a user buffer descriptor specifying a location of a plurality of user buffers in a first portion of memory that is allocated to an application program, wherein each user buffer is configured to store processed frame data for the delegated TCP connection; and indicating in a filed of the user buffer descriptor how many of the user buffers store the first processed frame data for the delegated TCP connection.

Connery discloses uploading the first processed frame data to a user buffer (fig. 4 element 111 discloses a buffer) in a first portion of memory that is allocated to an application program (fig. 4 indicates the buffer, 111, is located in a higher layer managed memory. Col. 6 lines 41-44 disclose it is an application managed memory.) and uploading the second processed frame data to a legacy buffer (fig. 4.110 discloses a buffer) in a second portion of memory that is allocated to a software driver (fig. 4 indicates the buffer, 110, is located in a driver managed memory) configured to communicate between the offload unit and a TCP stack (fig. 3 discloses the driver layer, 54, communicates between the offload unit, 56, and a TCP stack, 52.). Connery discloses passing identification of a target buffer as a buffer descriptor including an address and length in col. 5 lines 1-6.

It would have been obvious to one of ordinary skill in the art at the time of the invention to upload the first and second portion to buffers in application or driver memory respectively, as taught by Connery, in the offloading method taught by Hayes because doing so improves the performance and scalability of a network, as taught by Connery in col. 3 lines 4-10.

Allison discloses a buffer count field in a user buffer descriptor in fig. 3 and col. 5 lines 51-60. The field indicates the number of buffers which contain data.

It would have been obvious to one of ordinary skill in the art at the time of the invention to include a buffer descriptor with count field, as taught by Allison, in the offload method taught by Hayes because doing so helps provide architecture capable of speeds from 10-1000 Mbps, as taught by Allison in col. 2 lines 15-17.

In regards to claim 17, Hayes and Connery disclose the method of claim 1, wherein at least a portion of the first processed frame data is payload data (Connery fig. 4.102 discloses the first portion consists of payload data.

6. In regards to claims 19 and 20, Hayes discloses updating connection state information in paragraphs 57 and 67.

7. Claims 26, 28, and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hayes (US 2003/0158906) in view of Connery et al. (US 6,246,683) further in view of Allison et al. (US 6393,457) in view of known prior art.

8. In regards to claim 26, Hayes and Connery modified by Allison disclose the method of claim 1, but are silent further comprising determining that a sync request flag is not asserted, the sync request flag controlling flushing of existing user buffer descriptors for user buffers that do not store processed frame data and discarding of new user buffer descriptors for the delegated TCP connection that are received while the sync request flag is asserted.

Sync is a well known function in the art. The programming language UNIX, developed in 1969, includes a function titled sync. This function helps ensures multiple copies of a database are identical by flushing any existing but unused entries. While the sync function is

operating new information cannot be written to the database to prevent interference with the sync operation. A similar function is used in many other applications. For example, Microsoft Outlook 2001 includes a sync function. A user may access email from Outlook or a web interface from a desktop, laptop, or mobile device. Any changes made, regardless of application and device used, are maintained across all devices using the sync function. The sync function is also used by other applications such as an iPod and Windows Media Player, again to ensure copies of the database are the same at all locations. New entries are added and old, unused, or deleted entries are removed.

Official Notice is taken that it would have been obvious to one of ordinary skill in the art at the time of the invention to include a sync request flag, as was known in the art, in the offload method taught by Hayes and Connery modified by Allison because doing so ensures the database of buffers in use remains consistent between the main and offload units.

9. In regards to claim 28, Hayes and Connery modified by Allison disclose the system of claim 9, but are silent further comprising determining that a sync request flag is not asserted, the sync request flag controlling flushing of existing user buffer descriptors for user buffers that do not store processed frame data and discarding of new user buffer descriptors for the delegated TCP connection that are received while the sync request flag is asserted.

Sync is a well known function in the art. The programming language UNIX, developed in 1969, includes a function titled sync. This function helps ensures multiple copies of a database are identical by flushing any existing but unused entries. While the sync function is operating new information cannot be written to the database to prevent interference with the sync operation. A similar function is used in many other applications. For example, Microsoft

Outlook 2001 includes a sync function. A user may access email from Outlook or a web interface from a desktop, laptop, or mobile device. Any changes made, regardless of application and device used, are maintained across all devices using the sync function. The sync function is also used by other applications such as an iPod and Windows Media Player, again to ensure copies of the database are the same at all locations. New entries are added and old, unused, or deleted entries are removed.

Official Notice is taken that it would have been obvious to one of ordinary skill in the art at the time of the invention to include a sync request flag, as was known in the art, in the offload method taught by Hayes and Connery modified by Allison because doing so ensures the database of buffers in use remains consistent between the main and offload units.

10. In regards to claim 29, Hayes and Connery modified by Allison disclose the method of claim 16, but are silent further comprising determining that a sync request flag is not asserted, the sync request flag controlling flushing of existing user buffer descriptors for user buffers that do not store processed frame data and discarding of new user buffer descriptors for the delegated TCP connection that are received while the sync request flag is asserted.

Sync is a well known function in the art. The programming language UNIX, developed in 1969, includes a function titled sync. This function helps ensures multiple copies of a database are identical by flushing any existing but unused entries. While the sync function is operating new information cannot be written to the database to prevent interference with the sync operation. A similar function is used in many other applications. For example, Microsoft Outlook 2001 includes a sync function. A user may access email from Outlook or a web interface from a desktop, laptop, or mobile device. Any changes made, regardless of application

and device used, are maintained across all devices using the sync function. The sync function is also used by other applications such as an iPod and Windows Media Player, again to ensure copies of the database are the same at all locations. New entries are added and old, unused, or deleted entries are removed.

Official Notice is taken that it would have been obvious to one of ordinary skill in the art at the time of the invention to include a sync request flag, as was known in the art, in the offload method taught by Hayes and Connery modified by Allison because doing so ensures the database of buffers in use remains consistent between the main and offload units.

11. Claims 27 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hayes (US 2003/0158906) in view of Connery et al. (US 6,246,683) further in view of Allison et al. (US 6393,457) in view of Dunlap et al. (US 6,760,799).

12. In regards to claim 27, Hayes and Connery modified by Allison discloses the method of claim 1, but is silent further comprising: determining that a threshold value specified for a delegated TCP connection is exceeded; and setting a flag in a notification field indicating that the threshold value is exceeded for the delegated TCP connection.

Dunlap discloses monitoring a queue level and if the threshold is exceeded setting an "Interrupt Now" flag for notification in col. 7 lines 17-23.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the offload unit of Hayes to include the threshold and notification taught by Dunlap because doing so avoids multiple interrupts, as taught by Dunlap in col. 3 lines 10-22.

13. In regards to claim 30, Hayes and Connery modified by Allison discloses the method of claim 16, but is silent further comprising: determining that a threshold value specified for a

delegated TCP connection is exceeded; and setting a flag in a notification field indicating that the threshold value is exceeded for the delegated TCP connection.

Dunlap discloses monitoring a queue level and if the threshold is exceeded setting an "Interrupt Now" flag for notification in col. 7 lines 17-23.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the offload unit of Hayes to include the threshold and notification taught by Dunlap because doing so avoids multiple interrupts, as taught by Dunlap in col. 3 lines 10-22.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to KERRI M. ROSE whose telephone number is (571) 272-0542. The examiner can normally be reached on Monday through Thursday, 7:00 am - 4:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Aung MOE can be reached on (571) 272-7314. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Kerri M Rose/
Examiner, Art Unit 2416

/Aung S. Moe/
Supervisory Patent Examiner, Art Unit 2416